
SELF-SUPERVISED CONTRASTIVE LEARNING FOR VIDEOS USING DIFFERENTIABLE LOCAL ALIGNMENT

Keyne Oei

Universität des Saarlandes
Saarbrücken, Germany
s8keoeii@uni-saarland.de

Amr Gomaa

German Research Center for Artificial Intelligence (DFKI)
Saarbrücken, Germany
amr.gomaa@dfki.de

Anna Maria Feit

Universität des Saarlandes
Saarbrücken, Germany
feit@cs.uni-saarland.de

João Belo

Universität des Saarlandes
Saarbrücken, Germany
jbelo@cs.uni-saarland.de

March 4, 2025

ABSTRACT

Robust frame-wise embeddings are essential to perform video analysis and understanding tasks. We present a self-supervised method for representation learning based on aligning temporal video sequences. Our framework uses a transformer-based encoder to extract frame-level features and leverages them to find the optimal alignment path between video sequences. We introduce the novel Local-Alignment Contrastive (LAC) loss, which combines a differentiable local alignment loss to capture local temporal dependencies with a contrastive loss to enhance discriminative learning. Prior works on video alignment have focused on using global temporal ordering across sequence pairs, whereas our loss encourages identifying the best-scoring subsequence alignment. LAC uses the differentiable Smith-Waterman (SW) affine method, which features a flexible parameterization learned through the training phase, enabling the model to adjust the temporal gap penalty length dynamically. Evaluations show that our learned representations outperform existing state-of-the-art approaches on action recognition tasks.

1 Introduction

Video understanding and analysis have garnered significant attention in computer vision research in recent years [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. These tasks require models to capture not only spatial information within individual frames but also the complex temporal dynamics that evolve across sequences of frames. Traditional supervised learning methods have achieved considerable success in advancing performance; models such as those presented in [2, 4, 10] are able to predict action categories effectively. However, these methods present several challenges: (i) Most approaches require fine-grained annotations, which are time-consuming and labor-intensive; (ii) Most approaches struggle to capture the complex temporal information and causal relationships that vary across diverse videos. To address these challenges, we explore how representation learning based on the pretext-task of video alignment can improve performance in downstream action recognition tasks such as phase classification and phase progression.

Prior self-supervised or weakly-supervised video alignment approaches [11, 12, 13, 14, 15, 16] train on pairs of videos that describe the same action using either cycle-consistency [12], soft dynamic time warping [14, 13], or sequential contrastive learning [15]. We notice that the pretext task of video alignment shares similarities with bioinformatics sequence alignment for studying DNA, RNA, and protein structures. Since some actions are composed of sub-actions or events that occur in a specific order, video alignment shares similarities with protein alignment, as both involve identifying regions of continuity and discontinuity within sequences. There are two main types of alignment algorithms commonly used in bioinformatics: *global* and *local*. The Needleman-Wunsch (NW) algorithm [17], a global

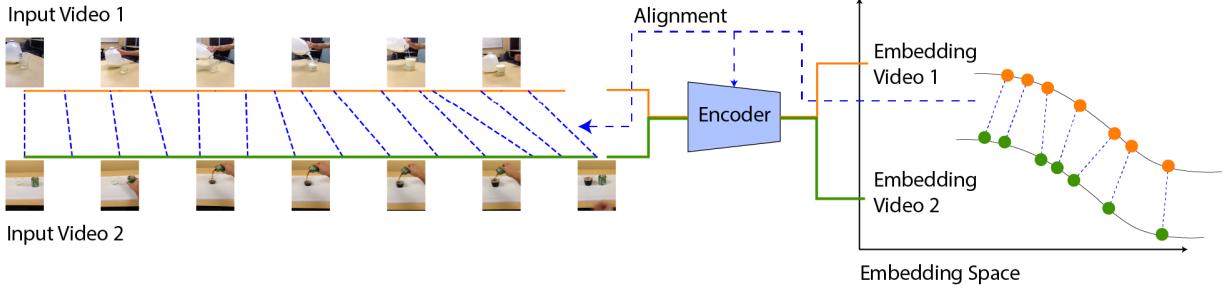


Figure 1: We introduce a representation learning approach that aligns video sequences depicting the same processes. Our training objective is to use a novel LAC loss to optimize and learn an element-wise embedding function that supports the alignment process.

alignment method, aligns entire sequences from start to finish. This approach is conceptually similar to Dynamic Time Warping (DTW) [18], which aligns sequences of varying lengths by matching their temporal patterns from start to finish. Differentiable versions of NW and DTW algorithms have been established by incorporating smooth maximum/minimum and argmax/argmin functions [19, 20]. In particular, Soft-DTW [20] has been used in prior video alignment approaches [13, 14, 16]. In contrast, the Smith-Waterman (SW) algorithm [21] implements local alignment, identifying regions of high similarity within sequences and allowing for gaps, insertions, and deletions. This ability to handle subsequences makes local alignment particularly suitable for modeling the complexity of actions in real-world video data, where actions may not always follow rigid, continuous sequences.

Specifically, in this work, we introduce a novel loss function termed *Local-Alignment Contrastive* (LAC) loss, integrated into our proposed end-to-end framework for the video alignment task, as visualized in Figure 1. Following prior works [15, 16], our framework uses a variation of convolutional and transformer encoders to extract spatio-temporal features from each frame. However, rather than relying on a differentiable global alignment loss like Soft-DTW [20], our novel LAC loss enables the comparison of video pairs through a consistent and differentiable local alignment loss. This loss specifically accommodates sequences with learned penalties for opening gaps and extending them, combined with a contrastive loss that effectively separates dissimilar frames. We demonstrate that the learned representations of our approach outperform previous methods on various downstream action recognition tasks, such as action phase classification and action phase progression on the Pouring [11] and PennAction [22] datasets. Code and models are available at: <https://github.com/keynekassapa13/LAC>.

2 Related Work

Action Recognition. Over the last decade, researchers have proposed various methods to address video recognition tasks [2, 4, 1, 3, 5], including action classification [23, 24, 2, 6, 8, 9], which involves mapping a video to an action category, and action segmentation [25, 26, 4, 27, 28], which aims to identify the spatial and temporal boundaries of specific actions within a video. However, these methods often rely heavily on extensive labeled data, which is time-consuming and labor-intensive to obtain. To address this limitation, self-supervised learning methods have been explored, enabling the extraction of video representations without requiring labeled data. Video alignment, as a self-supervised pretext task for action recognition, leverages temporal consistency to learn meaningful representations. Previous work has utilized traditional computer vision techniques to derive these representations; for instance, optical flow [29] computes motion between frames based on brightness constancy, the CONDENSATION algorithm [30] employs a probabilistic approach informed by prior motion data, and Space-Time Interest Points (STIPs) [31] detect salient points by analyzing both spatial and temporal patterns in video sequences.

Self-Supervised Learning. With the rise of deep learning, the field has shifted towards using Self-Supervised Learning (SSL) methods to learn video representations. Prior works [11, 12, 13, 14, 15] have addressed the problem of video alignment using self-supervised methods. Time Contrastive Network (TCN) [11] utilizes contrastive learning to distinguish frames from different segments while grouping those within the same segment. Temporal Cycle-Consistency (TCC) [12] introduces cycle-consistency loss to identify and match recurring action sequences within or across videos. Learning by Aligning Videos (LAV) [13] uses Soft-DTW [20] and Inverse Difference Moment (IDM) regularization to optimize alignment and ensure balanced frame distribution. Global Temporal Alignment (GTA) [14] implements a modified differentiable DTW with global consistency loss for consistent temporal alignment. Contrastive Action Representation Learning (CARL) [15] introduces Sequence Contrastive Loss (SCL), which minimizes the KL-

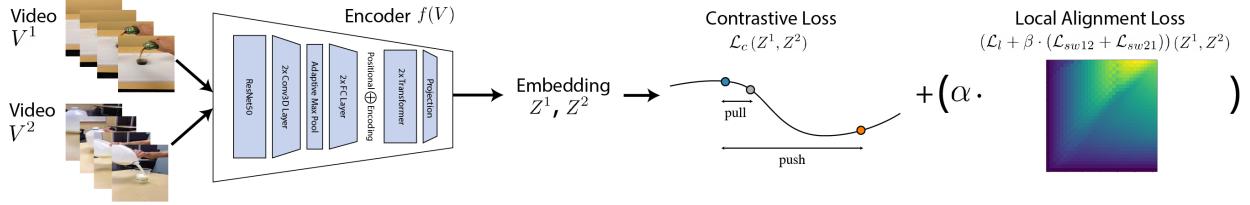


Figure 2: Our framework uses Local Alignment Loss and Contrastive Loss to optimize embeddings generated by an encoder that processes input videos after they have undergone spatio-temporal data augmentation.

divergence between augmented views based on timestamp distance. Learning Representation by position PROPropagation (LRProp) [16] utilizes a Soft-DTW [20] alignment path and a novel pair-wise position propagation technique. We will use these state-of-the-art methods to evaluate our model’s performance on action recognition tasks.

Global vs. Local Alignment. Global alignment methods, such as the NW algorithm [17] and DTW [18], aim to align entire sequences from start to finish to achieve the best overall match; differentiable versions have been developed [20, 19] and utilized in prior video alignment approaches [13, 14, 16]. In contrast, the SW algorithm [21] focuses on identifying the most similar subsequences within larger sequences. Recent work by Petti et al. [32] introduced a differentiable SW algorithm in a supervised framework that jointly learns alignments and improves protein structure prediction. [33] shows the power of embedding sequences into high-dimensional spaces within a supervised framework for more precise local alignment. Our approach employs a differentiable version of SW in self-supervised framework to improve model performance on video understanding and alignment tasks compared to state-of-the-art temporal global alignment methods.

3 Method

This section introduces our novel Local-Alignment Contrastive (LAC) loss function employed within our framework where video alignment serves as the pretext task, as visualized in Fig. 2. Specifically, we derive the forward and backward passes of the differentiable local alignment loss and explain how we ensure consistency within our method.

3.1 Notations

Let V^1 and V^2 represent two videos, each consisting of a sequence of frames. Specifically, $V^1 = \{v_t^1\}_{t=1}^{T_1}$ and $V^2 = \{v_t^2\}_{t=1}^{T_2}$, where v_t^1 and v_t^2 denote individual frames at time t , and T_1 and T_2 are the total number of frames in each video. Following [15], we preprocess the videos using temporal random cropping to generate cropped versions V_c^1 and V_c^2 . The indices of the sampled frames during cropping are recorded in S^1 and S^2 , where $S^1 \subseteq \{1, \dots, T_1\}$ and $S^2 \subseteq \{1, \dots, T_2\}$. Next, we apply data augmentations to the cropped videos, resulting in \tilde{V}^1 and \tilde{V}^2 . These augmented videos are then processed through an encoder to obtain embeddings Z^1 and Z^2 . Each embedding $Z^i \in \mathbb{R}^{T_i \times E}$ represents a sequence of length T_i with an embedding dimension E . The objective is to train the encoder to minimize the distance between embeddings Z^1 and Z^2 corresponding to videos depicting the same action, as determined by our loss.

3.2 Background

Prior approaches [13, 14] use DTW as the main method for the temporal alignment loss. Given embeddings Z^1 and Z^2 , we compute a cost matrix $C \in \mathbb{R}^{T_1 \times T_2}$, where each element $C_{i,j}$ measures the local distance between Z_i^1 and Z_j^2 . DTW seeks the optimal alignment path through this cost matrix that minimizes the total cumulative cost. The feasible paths must satisfy constraints of matching endpoints, monotonicity, and continuity. While the number of possible alignment paths grows exponentially with the lengths of the sequences, dynamic programming allows DTW to compute the optimal path efficiently in quadratic time $O(T_1 T_2)$. The DTW algorithm is formulated recursively as:

$$D_{i,j} = C_{i,j} + \min \begin{cases} D_{i-1,j-1} \\ D_{i-1,j} \\ D_{i,j-1} \end{cases} \quad (1)$$

where $D_{i,j}$ represents the accumulated cost up to point (i, j) , and $C_{i,j}$ is the local cost between elements Z_i^1 and Z_j^2 . The non-differentiable nature of the DTW formulation presents challenges for gradient-based optimization techniques. To address this, prior methods [13, 14] utilize Soft-DTW [20] which replaces the discrete min operator with a differentiable, smoothed version denoted as softmax^γ . As the smoothing parameter γ approaches zero, the behavior of softmax^γ approximates that of the discrete min operator. This substitution enables the use of gradient-based optimization in DTW while preserving its alignment capabilities.

3.3 Differentiable Local-Alignment

Despite the advantages offered by Soft-DTW [20], it inherently maintains the global alignment characteristic, ensuring continuity throughout the entire sequence. This attribute of global alignment may prove suboptimal in scenarios where localized alignment strategies, such as those implemented by the SW algorithm [21], are preferable. The SW algorithm [21] supports localized matching, providing enhanced flexibility and increased sensitivity to similarities within subsequences. Therefore, we propose a differentiable SW local-alignment approach as our method for temporal alignment loss.

We consider a pair of embeddings, (Z^1, S^1) and (Z^2, S^2) , with corresponding sampled indices, tasked with determining the optimal alignment between these embeddings. We begin by calculating a similarity matrix, $S \in \mathbb{R}^{T \times T}$, using the inverted Euclidean distance to yield higher values that indicate increased similarity between the points in Z^1 and Z^2 . Utilizing dynamic programming, the SW algorithm effectively determines the optimal alignment by maximizing a score derived from pairwise similarities within the matrix. The alignment algorithm is formally expressed as follows:

$$D_{i,j} = S_{i,j} + \max \begin{cases} 0 \\ D_{i-1,j-1} \\ I_{x_{i-1,j-1}} \\ I_{y_{i-1,j-1}} \end{cases} \quad (2)$$

$$I_{x_{i,j}} = \max \begin{cases} D_{i,j-1} - g_{o_{i,j}} \\ I_{x_{i,j-1}} - g_{e_{i,j}} \end{cases} \quad (3)$$

$$I_{y_{i,j}} = \max \begin{cases} D_{i-1,j} - g_{o_{i,j}} \\ I_{x_{i-1,j}} - g_{o_{i,j}} \\ I_{y_{i-1,j}} - g_{e_{i,j}} \end{cases} \quad (4)$$

where the initial values are set to $D(i, 0) = D(0, j) = I_x(i, 0) = I_x(0, j) = I_y(i, 0) = I_y(0, j) = -\infty$ for all indices $i, j = 1, \dots, T$. The matrix D described in Eq. 15 acts as the primary scoring matrix, representing the cumulative maximum score at each matrix coordinate (i, j) . The matrices I_x and I_y detailed in Eq. 16 and Eq. 4, represent the scores associated with introducing gaps along the x and y axes. Additionally, g_o and g_e are used to specify the penalties for opening and extending gaps.

Similar to Soft-DTW [20], our differentiable temporal alignment loss function uses a smoothed version of the discrete max function, denoted as \max^γ . This smooth approximation is formally defined as:

$$\max^\gamma(u_1, \dots, u_n) := \gamma \log \sum_{i=1}^n e^{u_i/\gamma} \quad (5)$$

where $\gamma > 0$ is a smoothing parameter that controls the approximation's fidelity to the original max function. As γ approaches zero, the smoothed \max^γ function increasingly approximates the behavior of the standard max function, enabling a differentiable formulation that can be integrated into gradient-based optimization frameworks. The gradient of the smoothed \max^γ with respect to each input u_i is calculated as follows:

$$\frac{\partial \max^\gamma(\{u_1, \dots, u_n\})}{\partial u_i} = \exp\left(\frac{u_i - \max^\gamma(\{u_1, \dots, u_n\})}{\gamma}\right) \quad (6)$$

This expression aligns with the softmax function. By differentiating the smoothed maximum score with respect to each input, our model directly utilizes derivative-sensitive parameters such as the similarity score (S), gap open penalty (g^o), and gap extend penalty (g^e). As a result, we define our differentiable local alignment loss as $\mathcal{L}_{sw_{ij}} = \text{SW}(\text{sim}(Z^i, Z^j))$.

The (sim) function calculates the similarity between two embeddings and generates a similarity matrix S . Unlike prior differentiable SW implementations [32, 33] that depend on supervised learning, our approach integrates the differentiable SW algorithm within a self-supervised framework.

3.4 Final Loss

Following the approach in [15], our contrastive loss is modeled after the NT-Xent loss from SimCLR [34]. This loss function calculates absolute pairwise distances between embeddings (Z^1, S^1) and (Z^2, S^2) , subsequently forming a Gaussian-weighted positive label distribution. This distribution is then contrasted against normalized similarity logits using the Kullback-Leibler divergence. The contrastive loss is mathematically expressed as:

$$\mathcal{L}_c = -\frac{1}{T} \sum_{i,j=1}^T \left(\frac{G(s_i^1 - s_j^2)}{\sum_{k=1}^T G(s_i^1 - s_k^2)} \log \frac{\exp(\text{sim}(z_i^1, z_j^2)/\tau)}{\sum_{k=1}^T \exp(\text{sim}(z_i^1, z_k^2)/\tau)} \right) \quad (7)$$

where τ represents the temperature parameter, G denotes the Gaussian weighting function applied to the absolute pairwise distances, and (sim) measures the normalized similarity between embeddings. To ensure consistency between contrastive and local alignment losses, we synchronize our temporal local loss by utilizing the primary score matrix D_{12} and D_{21} from \mathcal{L}_{sw12} and \mathcal{L}_{sw21} . Following the application of the softmax function to D , the resulting matrix is referred to as \tilde{D} . The logits matrix L is then derived by performing an element-wise multiplication of \tilde{D}_{12} and \tilde{D}_{21}^T . The cross-entropy loss for this logits matrix against the Gaussian-weighted labels is calculated as follows:

$$\mathcal{L}_l = -\frac{1}{T} \sum_{i,j=1}^T \left(\frac{G(s_i^1 - s_j^2)}{\sum_{k=1}^T G(s_i^1 - s_k^2)} \log \left(\frac{\exp(L_{ij})}{\sum_{k=1}^T \exp(L_{ik})} \right) \right),$$

where $\tilde{D}_{12,ij} = \frac{\exp(D_{12,ij}/\tau)}{\sum_{k=1}^T \exp(D_{12,ik}/\tau)}$, $\tilde{D}_{21,ij} = \frac{\exp(D_{21,ij}/\tau)}{\sum_{k=1}^T \exp(D_{21,ik}/\tau)}$, $L = \tilde{D}_{12} \cdot \tilde{D}_{21}^T$

Our proposed loss function, referred as Local-Alignment Contrastive (LAC), integrates contrastive and local alignment losses. It is formally defined as follows:

$$\mathcal{L} = \mathcal{L}_c + \alpha \cdot (\mathcal{L}_l + \beta \cdot (\mathcal{L}_{sw12} + \mathcal{L}_{sw21})) \quad (8)$$

where α and β are weights used to balance the components of the loss, set to 0.01 and 1, respectively.

4 Experiments

Datasets. We evaluate the performance of the LAC model on two datasets using a variety of evaluation metrics. The Pouring dataset [11], which focuses on the action of pouring liquids, includes 70 training and 14 testing videos. The PennAction dataset [22], featuring 13 human actions, contains 1140 training and 966 testing videos. We utilize the key events and phases for the videos in both datasets as proposed by TCC [12].

Implementation Details. Our encoder, $f : \mathbb{R}^{TxCxWxH} \rightarrow Z$, maps video inputs V into an embedding space Z . We use a ResNet50-v2 [35] as our backbone to extract features from the Conv4c layer with an output size of $10 \times 10 \times 512$. These features are then processed through adaptive max pooling, followed by two fully connected layers with ReLU activation. A subsequent linear layer projects the features into a 256-dimensional space. To enhance the model's capacity to capture long-range dependencies, we integrate sine-cosine positional encoding and employ a two-layer Transformer encoder. To improve the model's ability to capture long-range dependencies, we incorporate sine-cosine positional encoding and apply a two-layer Transformer encoder. The final embedding layer reduces the dimensionality to 128 for the frame-wise representations.

Evaluation Metrics. Following related work [12, 13, 14, 15, 16], we evaluate our model using the following metrics: (i) *Phase Classification*, which assesses the accuracy of action phase predictions by training an SVM classifier on our embeddings; (ii) *Phase Progression*, which evaluates how accurately our embeddings predict action progress using a linear regression model's average R-squared value, based on normalized timestamp differences; (iii) *Average Precision@K (AP@K)*, which evaluates fine-grained frame retrieval accuracy by calculating the proportion of correctly matched phase labels within the K closest frames; (iv) *Kendall's Tau*, which quantifies the temporal alignment between sequences by comparing the ratio of concordant to discordant frame pairs.

Method	Class			AP@K			Progress	τ
	10	50	100	K=5	K=10	K=15		
TCN [11]	80.32	81.44	83.56	76.26	76.71	77.26	82.30	83.51
TCC [12]	86.60	86.78	86.86	81.84	80.94	81.69	83.36	85.26
LAV [13]	89.77	90.35	91.77	87.48	88.36	88.40	85.20	88.75
GTA [14]	89.34	90.20	90.22	87.79	87.48	87.82	88.67	92.47
SCL [15]	92.76	92.80	93.05	88.75	88.51	88.97	91.26	98.20
LRPROP [16]	92.70	94.44	94.36	92.41	90.33	90.86	94.09	99.46
LAC	95.87	95.78	95.16	92.76	91.07	91.37	94.24	97.50

Table 1: Performance comparison of state-of-the-art methods on the Pouring Dataset [11]

Method	Class			AP@K			Progress	τ
	10	50	100	K=5	K=10	K=15		
TCN [11]	69.73	70.26	70.01	60.92	61.57	61.52	76.37	63.72
TCC [12]	86.60	86.78	86.86	81.84	80.94	81.69	83.36	85.26
LAV [13]	88.51	88.72	88.97	73.47	73.13	74.27	92.52	93.06
GTA [14]	84.21	84.68	85.28	71.72	72.17	71.52	90.51	83.35
SCL [15]	87.85	87.52	88.15	91.70	90.61	90.58	92.89	98.14
LRPROP [16]	91.90	92.96	93.25	92.46	92.2	92.03	93.03	99.09
LAC	95.57	93.79	93.40	93.87	93.41	92.65	94.21	94.10

Table 2: Performance comparison of state-of-the-art methods on the PennAction Dataset [22]

4.1 Results

We apply the same four metrics to the Pouring dataset [11]. For the PennAction dataset [22], we evaluate each of the 13 action categories using the same four metrics and report the averaged results across all categories. To ensure a fair comparison (e.g., consistent data splits, identical preprocessing), we replicated the evaluations of previous approaches using the GitHub repositories¹ provided by the original authors. Each model was trained using its respective pre-trained backbone. An exception was made for LRPROP, as their GitHub repository is not available.

Results on Pouring Dataset. Table 1 presents a comparison of our method’s performance against state-of-the-art approaches on the Pouring dataset. Bold and underlined text denote the best and second-best results. Notably, it achieves a +3.11% improvement on Phase Classification using only 10% of the labels. Additionally, our model excels in AP@K and Progress metrics. However, we observe lower performance in Kendall’s Tau. We hypothesize this is due to how the SW’s algorithm encourages skipping unnecessary segments of the sequence. The introduction of gap open and extend penalties could disrupt the continuity needed for high Kendall’s Tau scores.

Results on PennAction Dataset. Table 2 compares the performance of our method with state-of-the-art approaches on the PennAction dataset. Bold and underlined text denote the best and second-best results. LAC consistently outperforms previous methods across most metrics, with the exception of Kendall’s Tau. Notably, the improvement in AP@K is more pronounced on PennAction than on Pouring, likely due to the fewer number of frames in PennAction dataset, which may further impact alignment performance.

Qualitative analysis of results. Figure 3 illustrates the alignment process, with the optimal path depicted on the left and the frame-by-frame alignment on the right. This visualization demonstrates the synchronization of the two videos despite differences in their temporal progression and duration. Such alignment visualizations enable in-depth analysis of deviations or inefficiencies within specific actions in video understanding. Additionally, Figure 4 showcases our embedding-based retrieval system’s ability to accurately identify and retrieve frames corresponding to specific action sequences across different videos.

4.2 Ablation Study

This section presents multiple experiments on the *Pouring* dataset that analyze the different components of our framework.

¹github.com/google-research/google-research/tcc, github.com/trquhuytin/LAV-CVPR21, github.com/hadjisma/VideoAlignment, github.com/mingchen/CARL_code

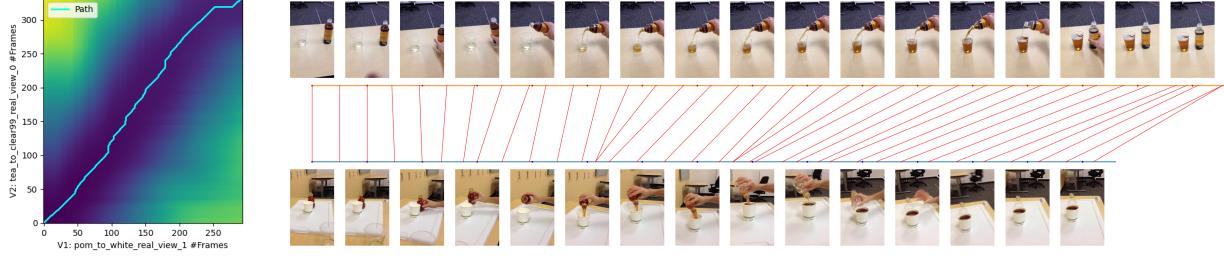


Figure 3: Similarity matrix (left) shows video alignment using an optimal path and the respective video aligned frame-by-frame (right).

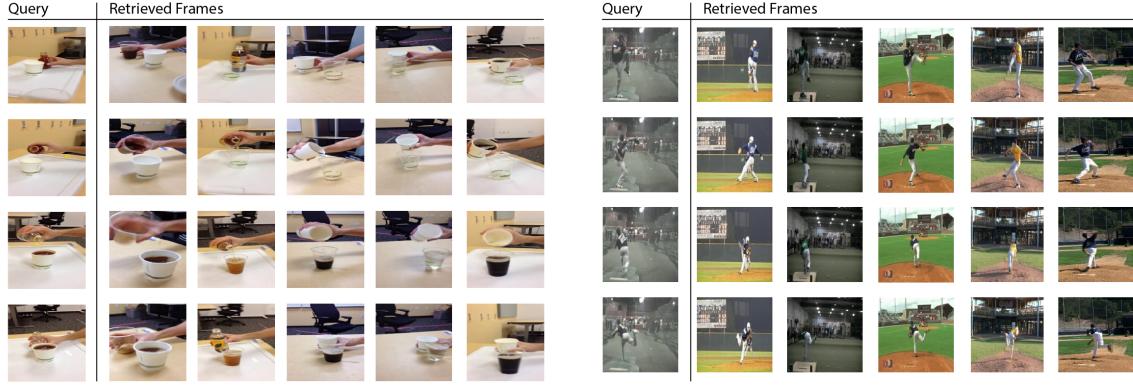


Figure 4: Fine-grained frame retrieval for Pouring (left) and PennAction (right) achieved using nearest neighbors within our embedding space.

Architecture	Class	Progress	τ
ResNet-50 + Convolutional 3D		88.04	73.26
ResNet-50 + Transformer (w/o pretrained weights)		90.06	89.32
ResNet-50 + Transformer	95.16	94.24	97.50

Table 3: Different Encoder Architecture on Model Performance.

γ	Class	Progress	τ
0.6	93.44	92.12	94.93
0.7	94.87	92.82	97.29
0.8	95.16	94.24	97.50
0.9	93.93	92.02	94.91

Table 4: Different γ Values on Model Performance

Loss \mathcal{L}	Class	Progress	τ
\mathcal{L}_c	93.44	92.12	94.93
$\mathcal{L}_c + \alpha \cdot \mathcal{L}_l$	93.71	92.16	95.01
$\mathcal{L}_c + \alpha \cdot (\mathcal{L}_l + \beta \cdot (\mathcal{L}_{sw12} + \mathcal{L}_{sw21}))$	95.16	94.24	97.50

Table 5: Different LAC Loss Formulations on Model Performance

Network Architecture. Table 3 demonstrates that our performance evaluation across various network architectures highlights the superiority of the ResNet-50 model with pretrained weights combined with a Transformer.

Hyperparameter of LAC Loss. Table 4 presents the optimal performance results obtained at the smoothing parameter $\gamma = 0.8$. Table 5 demonstrates that adding the local alignment loss (\mathcal{L}_l) to the contrastive loss (\mathcal{L}_c), the classification accuracy increases by 1.72, progression accuracy by 2.12, and the Tau score by 2.57.

5 Conclusion

The paper introduces a novel approach to representation learning through a Local-Alignment Contrastive (LAC) loss that integrates a differentiable local alignment loss with a contrastive loss, all within a self-supervised framework. The

method employs a differentiable affine Smith-Waterman algorithm to enable temporal alignment that dynamically adjusts to variations in action sequences. This approach is distinct in its focus on capturing local temporal dependencies and enhancing the discriminative learning of video embeddings, accommodating differences in action lengths and sequences. Experimental results on the Pouring and PennAction datasets showcase the method’s superior performance over existing state-of-the-art approaches.

References

- [1] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016.
- [2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.
- [3] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *Proceedings of the European conference on computer vision (ECCV)*, pages 803–818, 2018.
- [4] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019.
- [5] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 244–253, 2019.
- [6] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 203–213, 2020.
- [7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [8] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6836–6846, 2021.
- [9] Mengmeng Wang, Jiazheng Xing, and Yong Liu. Actionclip: A new paradigm for video action recognition. *arXiv preprint arXiv:2109.08472*, 2021.
- [10] Myeongjun Kim, Federica Spinola, Philipp Benz, and Tae-Hoon Kim. A*: Atrous spatial temporal action recognition for real time applications. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 7014–7024, 2024.
- [11] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE, 2018.
- [12] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1801–1810, 2019.
- [13] Sanjay Haresh, Sateesh Kumar, Huseyin Coskun, Shahram N Syed, Andrey Konin, Zeeshan Zia, and Quoc-Huy Tran. Learning by aligning videos in time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5548–5558, 2021.
- [14] Isma Hadji, Konstantinos G Derpanis, and Allan D Jepson. Representation learning via global temporal alignment and cycle-consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11068–11077, 2021.
- [15] Minghao Chen, Fangyun Wei, Chong Li, and Deng Cai. Frame-wise action representations for long videos via sequence contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13801–13810, 2022.
- [16] Guy Bar-Shalom, George Leifman, and Michael Elad. Weakly-supervised representation learning for video alignment and analysis. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6909–6919, 2024.
- [17] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.
- [18] Meinard Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

- [19] James T Morton, Charlie EM Strauss, Robert Blackwell, Daniel Berenberg, Vladimir Gligorijevic, and Richard Bonneau. Protein structural alignments from sequence. *BioRxiv*, pages 2020–11, 2020.
- [20] Marco Cuturi and Mathieu Blondel. Soft-dtw: a differentiable loss function for time-series. In *International conference on machine learning*, pages 894–903. PMLR, 2017.
- [21] Temple F Smith, Michael S Waterman, et al. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.
- [22] Weiyu Zhang, Menglong Zhu, and Konstantinos G Derpanis. From actemes to action: A strongly-supervised representation for detailed action understanding. In *Proceedings of the IEEE international conference on computer vision*, pages 2248–2255, 2013.
- [23] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.
- [24] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *Advances in neural information processing systems*, 27, 2014.
- [25] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2914–2923, 2017.
- [26] Kevin Duarte, Yogesh Rawat, and Mubarak Shah. Videocapsulenet: A simplified network for action detection. *Advances in neural information processing systems*, 31, 2018.
- [27] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. Bsn: Boundary sensitive network for temporal action proposal generation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.
- [28] Chen-Lin Zhang, Jianxin Wu, and Yin Li. Actionformer: Localizing moments of actions with transformers. In *European Conference on Computer Vision*, pages 492–510. Springer, 2022.
- [29] John L Barron, David J Fleet, and Steven S Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12:43–77, 1994.
- [30] Michael Isard and Andrew Blake. Condensation—conditional density propagation for visual tracking. *International journal of computer vision*, 29(1):5–28, 1998.
- [31] Ivan Laptev. On space-time interest points. *International journal of computer vision*, 64:107–123, 2005.
- [32] Samantha Petti, Nicholas Bhattacharya, Roshan Rao, Justas Dauparas, Neil Thomas, Juannan Zhou, Alexander M Rush, Peter Koo, and Sergey Ovchinnikov. End-to-end learning of multiple sequence alignments with differentiable smith–waterman. *Bioinformatics*, 39(1):btac724, 2023.
- [33] Felipe Llinares-López, Quentin Berthet, Mathieu Blondel, Olivier Teboul, and Jean-Philippe Vert. Deep embedding and alignment of protein sequences. *Nature methods*, 20(1):104–111, 2023.
- [34] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.
- [35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Supplementary Materials

A Architecture Details

The complete architecture of our encoder is shown in Table 6. The encoder consists of two main components: the *Spatial Encoder* and the *Temporal Encoder*. The *Spatial Encoder* is responsible for extracting spatial features from individual frames using ResNet-50 [35] with pretrained weights. This extraction process involves a series of convolutional layers, batch normalization, ReLU activations, max pooling, and bottleneck layers. Subsequently, the *Temporal Encoder* processes these spatial features across multiple frames to capture temporal dependencies. It employs fully connected layers with dropout and ReLU activations to enhance feature learning and prevent overfitting. A positional encoder is integrated to incorporate sequence information, and transformer encoder layers are utilized to model temporal relationships effectively. The final linear layer reduces the dimensionality of the embeddings to 128.

Model	Layer	Parameters	Output Size
Spatial Encoder	Conv2d	$7 \times 7, 64$	$112 \times 112 \times 64$
	BatchNorm2d + ReLU	-	
	MaxPool2d	3×3	$56 \times 56 \times 64$
	Bottleneck conv2_x	$1 \times 1, 64$ $3 \times 3, 64$ $1 \times 1, 256$	$56 \times 56 \times 256$
	Bottleneck conv3_x	$1 \times 1, 128$ $3 \times 3, 128$ $1 \times 1, 512$	$28 \times 28 \times 512$
	Bottleneck conv4_x	$1 \times 1, 256$ $3 \times 3, 256$ $1 \times 1, 1024$	$14 \times 14 \times 1024$
	Conv3d	$3 \times 3 \times 3, 512$	$2 \times 10 \times 10 \times 512$
	BatchNorm3d + ReLU	-	
	AdaptiveMaxPool3d	1	$1 \times 1 \times 1 \times 512$
Temporal Encoder	Dropout	$p = 0.1$	64×512
	Linear	512	
	ReLU	-	
	Linear	256	64×256
	Positional Encoder	-	$2 \times 32 \times 256$
	Transformer Encoder	$256, 8, 1024 \times 2$	$2 \times 32 \times 256$
	Linear	128	64×128

Table 6: Summarized Composition of the Spatial and Temporal Encoder Architectures. The parameter structure is as follows: For convolution layers, the format is kernel size and number of output channels (e.g., $7 \times 7, 64$). For pooling layers, the format is kernel size (e.g., 3×3). For other layers, specific parameters are listed (e.g., $p = 0.1$ for dropout). For the Transformer Encoder, the format is the dimension of input and output vectors, number of attention heads, and dimension of the feedforward network hidden layer (e.g., $[256, 8, 1024] \times 2$).

B Differentiable Local-Alignment in Detail

We provide a detailed derivation of the backward recursion used in differentiable local alignment algorithms. Following the approach described in [33], we transform the discrete max function into a differentiable form, specifically a smooth max function. This smooth max function is mathematically represented as follows:

$$\max^\gamma(u_1, \dots, u_n) := \gamma \log \sum_{i=1}^n e^{u_i/\gamma}, \quad (9)$$

Equation 9 enables differentiation that results in a softmax function, detailed as follows:

$$\frac{\partial \max^\gamma(\{u_1, \dots, u_n\})}{\partial u_i} = \exp\left(\frac{u_i - \max^\gamma(\{u_1, \dots, u_n\})}{\gamma}\right) \quad (10)$$

$$= \exp\left(\frac{u_i - \gamma \log\left(\sum_{j=1}^n \exp\left(\frac{u_j}{\gamma}\right)\right)}{\gamma}\right) \quad (11)$$

$$= \exp\left(\frac{u_i}{\gamma} - \log\left(\sum_{j=1}^n \exp\left(\frac{u_j}{\gamma}\right)\right)\right) \quad (12)$$

$$= \exp\left(\frac{u_i}{\gamma}\right) \cdot \exp\left(-\log\left(\sum_{j=1}^n \exp\left(\frac{u_j}{\gamma}\right)\right)\right) \quad (13)$$

$$= \frac{\exp\left(\frac{u_i}{\gamma}\right)}{\sum_{j=1}^n \exp\left(\frac{u_j}{\gamma}\right)} \quad (14)$$

This transformation makes the function differentiable, allowing the use of gradient-based optimization in alignment algorithms. The Smith-Waterman (SW) local alignment algorithm [21] can be defined using dynamic programming as follows:

$$D_{i,j} = S_{i,j} + \max \begin{cases} 0 \\ D_{i-1,j-1} \\ I_{x_{i-1,j-1}} \\ I_{y_{i-1,j-1}} \end{cases} \quad (15)$$

$$I_{x_{i,j}} = \max \begin{cases} D_{i,j-1} - g_{o_{i,j}} \\ I_{x_{i,j-1}} - g_{e_{i,j}} \end{cases}, \quad I_{y_{i,j}} = \max \begin{cases} D_{i-1,j} - g_{o_{i,j}} \\ I_{x_{i-1,j}} - g_{o_{i,j}} \\ I_{y_{i-1,j}} - g_{e_{i,j}} \end{cases} \quad (16)$$

where $D_{i,j}$ represents the score of the optimal alignment ending at positions i and j in the two sequences being aligned. The scores for initiating or extending a gap are handled by $I_{x_{i,j}}$ and $I_{y_{i,j}}$. Additionally, $g_{o_{i,j}}$ and $g_{e_{i,j}}$ denote the penalties for opening and extending gaps. This method enables differentiation of the smoothed maximum score, denoted as $s^* = \max^\gamma(0, \max^\gamma(D))$, with respect to S , g_o , and g_e , treating these as parameters sensitive to derivatives. The differentiation of the smoothed maximum score with respect to S can be defined as follows:

$$\frac{\partial s^*}{\partial S_{i,j}} = \left(\frac{\partial s^*}{\partial D_{i,j}} \right) \frac{\partial D_{i,j}}{\partial S_{i,j}} \quad (17)$$

Using the inverse relationship defined in Eq. 16, the differentiation of the smoothed maximum score with respect to g_o is described as follows:

$$\frac{\partial s^*}{\partial g_{o_{i,j}}} = \frac{\partial s^*}{\partial I_{x_{i,j}}} \frac{\partial I_{x_{i,j}}}{\partial g_{o_{i,j}}} + \frac{\partial s^*}{\partial I_{y_{i,j}}} \frac{\partial I_{y_{i,j}}}{\partial g_{o_{i,j}}} \quad (18)$$

$$= \frac{\partial s^*}{\partial I_{x_{i,j}}} \frac{\partial \max^\gamma(D_{i,j-1} - g_{o_{i,j}}, I_{x_{i,j-1}} - g_{e_{i,j}})}{\partial g_{o_{i,j}}} \quad (19)$$

$$+ \frac{\partial s^*}{\partial I_{y_{i,j}}} \frac{\partial \max^\gamma(D_{i-1,j} - g_{o_{i,j}}, I_{x_{i-1,j}} - g_{o_{i,j}}, I_{y_{i-1,j}} - g_{e_{i,j}})}{\partial g_{o_{i,j}}} \quad (20)$$

$$= \frac{\partial s^*}{\partial I_{x_{i,j}}} \left(-\frac{\partial I_{x_{i,j}}}{\partial D_{i,j-1}} \right) + \frac{\partial s^*}{\partial I_{y_{i,j}}} \left(-\frac{\partial I_{y_{i,j}}}{\partial D_{i-1,j}} - \frac{\partial I_{y_{i,j}}}{\partial I_{x_{i-1,j}}} \right) \quad (21)$$

Similar to the above, using the inverse relationship from Eq. 16, the differentiation of the smoothed maximum score with respect to g_e is defined as follows:

$$\frac{\partial s^*}{\partial g_{e_{i,j}}} = \frac{\partial s^*}{\partial I_{x_{i,j}}} \frac{\partial I_{x_{i,j}}}{\partial g_{e_{i,j}}} + \frac{\partial s^*}{\partial I_{y_{i,j}}} \frac{\partial I_{y_{i,j}}}{\partial g_{e_{i,j}}} \quad (22)$$

$$= \frac{\partial s^*}{\partial I_{x_{i,j}}} \frac{\partial \max^\gamma(D_{i,j-1} - g_{o_{i,j}}, I_{x_{i,j-1}} - g_{e_{i,j}})}{\partial g_{e_{i,j}}} \quad (23)$$

$$+ \frac{\partial s^*}{\partial I_{y_{i,j}}} \frac{\partial \max^\gamma(D_{i-1,j} - g_{o_{i,j}}, I_{x_{i-1,j}} - g_{o_{i,j}}, I_{y_{i-1,j}} - g_{e_{i,j}})}{\partial g_{e_{i,j}}} \quad (24)$$

$$= \frac{\partial s^*}{\partial I_{x_{i,j}}} \left(-\frac{\partial I_{x_{i,j}}}{\partial I_{x_{i,j-1}}} \right) + \frac{\partial s^*}{\partial I_{y_{i,j}}} \left(-\frac{\partial I_{y_{i,j}}}{\partial I_{y_{i-1,j}}} \right) \quad (25)$$

Following [33], we consider backtracking the SW algorithm to set the recursive calculation and see how changes in the current state propagate gradients forward to influence future states. We begin by deriving the expression for $\frac{\partial s^*}{\partial D_{i,j}}$ from Eq. 17, which can be obtained as follows:

$$\frac{\partial s^*}{\partial D_{i,j}} = \frac{\partial s^*}{\partial D_{i,j}} + \frac{\partial s^*}{\partial D_{i+1,j+1}} \frac{\partial D_{i+1,j+1}}{\partial D_{i,j}} + \frac{\partial s^*}{\partial I_{x_{i,j+1}}} \frac{\partial I_{x_{i,j+1}}}{\partial D_{i,j}} + \frac{\partial s^*}{\partial I_{y_{i+1,j}}} \frac{\partial I_{y_{i+1,j}}}{\partial D_{i,j}} \quad (26)$$

In this equation, $\frac{\partial s^*}{\partial D_{i,j}}$ represents the immediate partial derivative of s^* with respect to $D_{i,j}$, reflecting its direct impact. The term $\frac{\partial s^*}{\partial D_{i+1,j+1}} \frac{\partial D_{i+1,j+1}}{\partial D_{i,j}}$ accounts for how $D_{i,j}$ affects the future state $D_{i+1,j+1}$. Similarly, $\frac{\partial s^*}{\partial I_{x_{i,j+1}}} \frac{\partial I_{x_{i,j+1}}}{\partial D_{i,j}}$ reflects the influence on the future state $I_{x_{i,j+1}}$, and $\frac{\partial s^*}{\partial I_{y_{i+1,j}}} \frac{\partial I_{y_{i+1,j}}}{\partial D_{i,j}}$ reflects the influence on the future state $I_{y_{i+1,j}}$.

From Equations 21 and 25, we see that the remaining partial derivatives to be calculated are $\frac{\partial s^*}{\partial I_{x_{i,j}}}$ and $\frac{\partial s^*}{\partial I_{y_{i,j}}}$. These derivatives can be derived recursively as follows:

$$\frac{\partial s^*}{\partial I_{y_{i,j}}} = \frac{\partial s^*}{\partial D_{i+1,j+1}} \frac{\partial D_{i+1,j+1}}{\partial I_{y_{i,j}}} + \frac{\partial s^*}{\partial I_{y_{i+1,j}}} \frac{\partial I_{y_{i+1,j}}}{\partial I_{y_{i,j}}} \quad (27)$$

$$\frac{\partial s^*}{\partial I_{x_{i,j}}} = \frac{\partial s^*}{\partial D_{i+1,j+1}} \frac{\partial D_{i+1,j+1}}{\partial I_{x_{i,j}}} + \frac{\partial s^*}{\partial I_{x_{i,j+1}}} \frac{\partial I_{x_{i,j+1}}}{\partial I_{x_{i,j}}} + \frac{\partial s^*}{\partial I_{y_{i+1,j}}} \frac{\partial I_{y_{i+1,j}}}{\partial I_{x_{i,j}}} \quad (28)$$

where $\frac{\partial s^*}{\partial I_{y_{i,j}}}$ accounts for the influence of $I_{y_{i,j}}$ on the future states $D_{i+1,j+1}$ and $I_{y_{i+1,j}}$. Similarly, $\frac{\partial s^*}{\partial I_{x_{i,j}}}$ reflects the influence of $I_{x_{i,j}}$ on the future states $D_{i,j+1}$ and $I_{x_{i,j+1}}$. By recursively computing these partial derivatives, we can capture how changes in the current states $D_{i,j}$, $I_{x_{i,j}}$, and $I_{y_{i,j}}$ propagate through the dynamic programming matrix.

C More results

In the main paper, the main result showcases the performance comparison of LAC and SOTA on the PennAction Dataset [22]. The results represent the average across all 13 actions. Details of these actions can be found in Table 7.

Action	Class			AP@K			Progress	τ
	10	50	100	K=5	K=10	K=15		
baseball_pitch	92.50	90.95	90.68	91.48	91.41	90.95	93.28	88.64
baseball_swing	93.96	91.85	90.36	93.51	93.23	91.8	94.26	95.55
bench_press	94.83	91.64	91.17	93.76	92.88	91.53	96.64	93.42
bowl	96.83	94.53	93.49	92.52	91.96	90.54	86.32	91.84
clean_and_jerk	92.71	88.83	87.07	87.36	86.82	85.64	92.69	94.66
golf_swing	96.37	93.76	93.5	93.28	92.92	91.82	95.86	94.83
jumping_jacks	95.30	94.3	93.87	94.36	95.31	94.66	95.68	96.47
pushup	94.04	94.84	94.38	94.66	94.38	94.23	96.8	96.64
pullup	95.56	94.55	94.26	94.57	94.87	94.12	96.4	95.44
situp	98.52	96.57	96.6	95.22	94.98	94.95	96.25	96.31
squat	97.65	94.28	96.21	95.26	95.52	94.64	93.91	95.08
tennis_forehand	97.51	97.06	97.51	96.55	95.03	94.46	94.29	92.05
tennis.Serve	96.62	96.09	95.09	97.76	95.01	95.10	92.33	92.35
average	95.57	93.79	93.40	93.87	93.41	92.65	94.21	94.10

Table 7: Performance comparison of LAC and SOTA on the PennAction Dataset [22] across 13 actions.